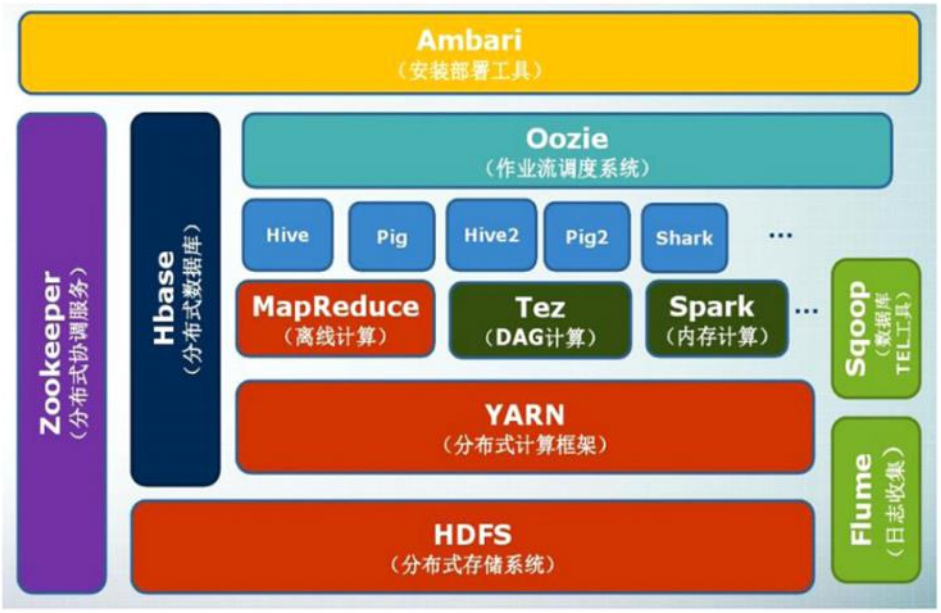
Hadoop入门

2020年8月21日 14:17

1. 简介
    a. 是Apache软件基金会旗下的一个开源分布式计算平台，为用户提供了系统底层细节透明的分布式基础架构
        i. Hadoop最初是由Apache Lucene项目的创始人Doug Cutting开发的文本搜索库。Hadoop源自始于2002年的Apache Nutch项目——一个开源的网络搜索引擎并且也是Lucene项目的一部分
        ii. 在2004年，Nutch项目也模仿GFS开发了自己的分布式文件系统NDFS（Nutch Distributed File System），也就是HDFS的前身
        iii. 2004年，谷歌公司又发表了另一篇具有深远影响的论文，阐述了MapReduce分布式编程思想
        iv. 2005年，Nutch开源实现了谷歌的MapReduce
        v. 到了2006年2月，Nutch中的NDFS和MapReduce开始独立出来，成为Lucene项目的一个子项目，称为Hadoop，同时，Doug Cutting加盟雅虎
        vi. 2008年1月，Hadoop正式成为Apache顶级项目，Hadoop也逐渐开始被雅虎之外的其他公司使用
        vii. 2008年4月，Hadoop打破世界纪录，成为最快排序1TB数据的系统，它采用一个由910个节点构成的集群进行运算，排序时间只用了209秒
        viii. 在2009年5月，Hadoop更是把1TB数据排序时间缩短到62秒。Hadoop从此名声大震，迅速发展成为大数据时代最具影响力的开源分布式开发平台，并成为事实上的大数据处理标准
    b. 基于Java语言开发的，具有很好的跨平台特性，并且可以部署在廉价的计算机集群中
    c. 核心是分布式文件系统HDFS（Hadoop Distributed File System）和数据计算作业MapReduce
    d. 被公认为行业大数据标准开源软件，在分布式环境下提供了海量数据的处理能力
    e. 几乎所有主流厂商都围绕Hadoop提供开发工具、开源软件、商业化工具和技术服务，如谷歌、雅虎、微软、思科、淘宝等
    f. 运行模式
        i. 独立式：Hadoop运行所有的东西在无后台的单独的JVM中，单进程单机模式
        ii. 伪分布式：Hadoop做为后台应用运行在一台机器，模拟小集群
        iii. 全分布式：Hadoop做为后台应用运行真实的集群多台电脑中
2. 组件

a. 

b.

组件	功能
HDFS	分布式文件系统
MapReduce	分布式并行编程模型
YARN	资源管理和调度器
Tez	运行在YARN之上的下一代Hadoop查询处理框架
Hive	Hadoop上的数据仓库
HBase	Hadoop上的非关系型的分布式数据库
Pig	一个基于Hadoop的大规模数据分析平台，提供类似SQL的查询语言Pig Latin
Sqoop	用于在Hadoop与传统数据库之间进行数据传递
Oozie	Hadoop上的工作流管理系统
Zookeeper	提供分布式协调一致性服务
Storm	流计算框架
Flume	一个高可用的，高可靠的，分布式的海量日志采集、聚合和传输的系统
Ambari	Hadoop快速部署工具，支持Apache Hadoop集群的供应、管理和监控
Kafka	一种高吞吐量的分布式发布订阅消息系统，可以处理消费者规模的网站中的所有动作流数据
Spark	类似于Hadoop MapReduce的通用并行框架

c. 一些免费的Hadoop发行版
   i. Apache（最原始的版本，所有发行版均基于这个版本进行改进）
   ii. Cloudera版本（Cloudera’s Distribution Including Apache Hadoop，简称“CDH”）http://archive.cloudera.com/cdh5/cdh/5/
   iii. Hortonworks版本（Hortonworks Data Platform，简称“HDP”）

3. 节点
   a. NameNode：负责协调集群中的数据存储，记录各文件各块所在的数据节点位置

的映射关系

i. FsImage：维护文件系统树以及文件树中所有的文件和文件夹的元数据
ii. 操作日志文件EditLog：记录所有针对文件的创建、删除、重命名等操作
iii. 当EditLog文件非常大的时候，会导致名称节点启动操作非常慢，而在这段时间内HDFS系统处于安全模式，一直无法对外提供写操作，影响了用户的使用

b. SecondaryNameNode：帮助NameNode收集文件系统运行的状态信息，对**元数据信息的备份，减少名称节点重启的时间**
   i. SecondaryNameNode会定期和NameNode通信，请求其停止使用EditLog文件，暂时将新的写操作写到一个新的文件edit.new上来，这个操作是瞬间完成，上层写日志的函数完全感觉不到差别
   ii. SecondaryNameNode通过HTTP GET方式从NameNode上获取到FsImage和EditLog文件，并下载到本地的相应目录下
   iii. SecondaryNameNode将下载下来的FsImage载入到内存，然后一条一条地执行EditLog文件中的各项更新操作，使得内存中的FsImage保持最新；这个过程就是EditLog和FsImage文件合并
   iv. 再通过post方式将新的FsImage文件发送到NameNode节点上
   v. NameNode将从SecondaryNameNode接收到的新的FsImage替换旧的FsImage文件，同时将edit.new替换EditLog文件，通过这个过程EditLog就变小了
c. DataNode：在磁盘存储被拆分的数据块
   i. 每个数据节点中的数据会被保存在各自节点的本地Linux文件系统中
d. JobTracker：协调数据计算任务（旧版）
e. TaskTracker：负责执行由JobTracker指派的任务（旧版）
f. YARN：资源管理框架（2.0版）

4. 存储原理
   a. 冗余数据保存：通常一个数据块的多个副本会被分布到不同的数据节点上
      i. 加快数据传输速度
      ii. 容易检查数据错误
      iii. 保证数据可靠性
   b. 数据存取策略
      i. 存：
         1) 第一个副本：放置在上传文件的数据节点；如果是集群外提交，则随机挑选一台磁盘不太满、CPU不太忙的节点
         2) 第二个副本：放置在与第一个副本不同的机架的节点上
         3) 第三个副本：与第一个副本相同机架的其他节点上
         4) 更多副本：随机节点
      ii. 取：
         1) HDFS提供了一个API可以确定一个数据节点所属的机架ID，客户端也可以调用API获取自己所属的机架ID
         2) 当客户端读取数据时，从名称节点获得数据块不同副本的存放位置列

表，列表中包含了副本所在的数据节点，可以调用API来确定客户端和这些数据节点所属的机架ID，优先选择客户端对应的机架的副本，如果没有，就随机选择一个副本读取数据

c. 数据错误与恢复：兼容廉价的硬件，把硬件出错看作一种常态
   i. 名称节点出错：根据备份服务器SecondaryNameNode中的FsImage和Editlog数据进行恢复
   ii. 数据节点出错
      1) 每个数据节点会定期向名称节点发送“心跳”信息，向名称节点报告自己的状态
      2) 当数据节点发生故障，或者网络发生断网时，名称节点就无法收到来自一些数据节点的心跳信息，这时，这些数据节点就会被标记为“宕机”，节点上面的所有数据都会被标记为“不可读”，名称节点不会再给它们发送任何I/O请求
      3) 这时，有可能出现一种情形，即由于一些数据节点的不可用，会导致一些数据块的副本数量小于冗余因子
      4) 名称节点会定期检查这种情况，一旦发现某个数据块的副本数量小于冗余因子，就会启动数据冗余复制，为它生成新的副本
      5) HDFS和其它DFS的最大区别就是**可以调整冗余数据的位置**
   iii. 数据出错：网络传输和磁盘错误等因素造成的数据错误
      1) 客户端在读取到数据后，会采用md5和sha1对数据块进行校验，以确定读取到正确的数据
      2) 在文件被创建时，客户端就会对每一个文件块进行信息摘录，并把这些信息写入到同一个路径的隐藏文件里面
      3) 当客户端读取文件的时候，会先读取该信息文件，然后，利用该信息文件对每个读取的数据块进行校验，如果校验出错，客户端就会请求到另外一个数据节点读取该文件块，并且向名称节点报告这个文件块有错误，名称节点会定期检查并且重新复制这个块

5. HDFS Shell命令
   a. hadoop fs适用于任何不同的文件系统，比如本地文件系统和HDFS文件系统
   b. hadoop dfs = hdfs dfs 只能适用于HDFS文件系统
      i. hadoop dfs -ls /
      ii. mkdir(创建目录)
         1) hadoop dfs -mkdir [-p]（p和Linux系统效果一样)
         2) hadoop dfs -mkdir -p /test/test1/test/test2
      iii. mv
         1) hadoop dfs -mv URI [URI ...]
         2) hadoop dfs –mv /output/CHANGES.txt/output/README.md/test/test2/
      iv. put(上传)
         1) hadoop dfs -put ...

2) hadoop dfs-put c.txt d.txt/test/test2/

v. rm（删除）

1) hadoop dfs -rm [-f] [-r|-R] [-skipTrash] URI [URI ...] (f：是否确认 r:递归删除 skipTrash：直接删除)

2) hadoop dfs-rm-r/test/

vi. 更多命令示例，查看帮助 打命令： hadoop dfs

6. Java API

a. Configuration类：该类的对象封转了客户端或者服务器的配置

b. FileSystem类：该类的对象是一个文件系统对象，可以用该对象的一些方法来对文件进行操作。FileSystem fs = FileSystem.get(conf);通过FileSystem的静态方法get获得该对象

c. FSDataInputStream和FSDataOutputStream：这两个类是HDFS中的输入输出流。分别通过FileSystem的open方法和create方法获得

```
 //创建目录
public static void mkdir(String path) throws IOException{
Configuration conf = new Configuration();
 FileSystem fs = FileSystem.get(conf);
 Path srcPath = new Path(path);
 boolean isok = fs.mkdirs(srcPath);
 if(isok){
 System.out.println("create dir ok!");
  }else{
     System.out.println("create dir failure");
   }
  fs.close();
  }
     //读取文件的内容
    public static void readFile(String filePath) throws IOException{
    Configuration conf = new Configuration();
    FileSystem fs = FileSystem.get(conf);
    Path srcPath = new Path(filePath);
   InputStream in = null;
   try {
   in = fs.open(srcPath);
      IOUtils.copyBytes(in, System.out, 4096, false); //复制到标准输出流
   } finally {
       IOUtils.closeStream(in);
  }  }
```

7. 局限

a. Hadoop1.0的核心组件（仅指MapReduce和HDFS，不包括Hadoop生态系统内的Pig、Hive、HBase等其他组件），主要存在以下不足：

i. 抽象层次低，需人工编码

ii. 表达能力有限

iii. 开发者自己管理作业（Job）之间的依赖关系

iv. 难以看到程序整体逻辑

v. 执行迭代操作效率低

vi. 资源浪费（Map和Reduce分两阶段执行）

vii. 实时性差（适合批处理，不支持实时交互式）

b.

组件	Hadoop1.0的问题	Hadoop2.0的改进
HDFS	单一名称节点，存在单点失效问题	设计了HDFS HA，提供名称节点热备机制

b.

HDFS	单一命名空间，无法实现资源隔离	设计了HDFS Federation，管理多个命名空间
MapReduce	资源管理效率低	设计了新的资源管理框架YARN

c. HA（High Availability）是为了解决单点故障问题
   i. 设置两个名称节点，“活跃（Active）”和“待命（Standby）”
   ii. 两种名称节点的状态同步，可以借助于一个共享存储系统来实现
   iii. 一旦活跃名称节点出现故障，就可以立即切换到待命名称节点
   iv. Zookeeper确保一个名称节点在对外服务
   v. 名称节点维护映射信息，数据节点同时向两个名称节点汇报信息
   vi. 是**热备份**，不需停机维护
d. Federation进一步解决了
   i. HDFS集群扩展性：多个名称节点各自分管一部分目录，使得一个集群可以扩展到更多节点，不再像HDFS1.0中那样由于内存的限制制约文件存储数目
   ii. 性能更高效：多个名称节点管理不同的数据，且同时对外提供服务，将为用户提供更高的读写吞吐率
   iii. 良好的隔离性：用户可根据需要将不同业务数据交由不同名称节点管理，这样不同业务之间影响很小
e. Federation的设计
   i. 在HDFS Federation中，设计了多个相互独立的名称节点，使得HDFS的命名服务能够水平扩展，这些名称节点分别进行各自命名空间和块的管理，不需要彼此协调，相互之间是（Federation）联盟关系。并且向后兼容
   ii. HDFS Federation中，所有名称节点会共享底层的数据节点存储资源，数据节点向所有名称节点汇报
   iii. 属于同一个命名空间的块构成一个“块池”
   iv. 对于Federation中的多个命名空间，可以采用客户端挂载表（Client Side Mount Table）方式进行数据共享和访问
   v. 客户可以访问不同的挂载点来访问不同的子命名空间
   vi. 把各个命名空间挂载到全局“挂载表”（mount-table）中，实现数据全局共享
   vii. 同样的命名空间挂载到个人的挂载表中，就成为应用程序可见的命名空间

f.

组件	功能	解决Hadoop中存在的问题
Pig	处理大规模数据的脚本语言，用户只需要编写几条简单的语句，系统会自动转换为MapReduce作业	抽象层次低，需要手工编写大量代码
Spark	基于内存的分布式并行编程框架，具有较高的实时性，并且较好支持迭代计算	延迟高，而且不适合执行迭代计算
Oozie	工作流和协作服务引擎，协调Hadoop上运行的不同任务	没有提供作业（Job）之间依赖关系管理机制，需要用户自己处理作业之间依赖关系

Tez	支持DAG作业的计算框架，对作业的操作进行重新分解和组合，形成一个大的DAG作业，减少不必要操作	不同的MapReduce任务之间存在重复操作，降低了效率
Kafka	分布式发布订阅消息系统，一般作为企业大数据分析平台的数据交换枢纽，不同类型的分布式系统可以统一接入到Kafka，实现和Hadoop各个组件之间的不同类型数据的实时高效交换	Hadoop生态系统中各个组件和其他产品之间缺乏统一的、高效的数据交换中介